# GCCS/DII COE System Integration Support

# Technical Report/Study: Developer Security Guidelines for GCCS COE

**Version 1.0**

February 20, 1997

Prepared for:

DISA/JEJA
ATTN: Ms. Claire Burchell
45335 Vintage Park Plaza
Sterling, VA 20166-6701

Contract Number: DCA100-94-D-0014
Delivery Order Number: 330, Task 2
CDRL Number: A005

Prepared by:

Computer Sciences Corporation
Defense Enterprise Integration Services
Four Skyline Drive
5113 Leesburg Pike, Suite 700
Falls Church, VA 22041

**THIS DOCUMENT IS UNCLASSIFIED**

## TABLE OF CONTENTS

## 1.0 INTRODUCTION

The Global Command and Control System (GCCS) Developer Security Guidelines provide information to the developer to ensure that new COE kernel, application, and database segments do not disrupt the security configuration for a GCCS implementation. The security configuration for GCCS is based on the GCCS Security Checklist, which can be found in Appendix B of the Trusted Facility Manual (TFM), that provides the foundation criteria for certification and accreditation (C&A). The Developer Security Guidelines may be updated as the GCCS Security Checklist and the C&A requirements evolve.

The security configuration for the GCCS is evolving to become an even stronger barrier against security threats ranging from the accidental actions by an authorized user to the premeditated actions of a determined hacker. This requires thoughtful consideration not only in the configuration of the system but also from the software developer to ensure that vulnerabilities that are developed-in are held to a minimum. Indeed, the software development process is the single-most important control aspect for enhancing or undermining the capability of creating and maintaining a security configuration. The developers of applications and database software simply must be made aware of the implications of poorly engineered software and its affect on the task of creating and maintaining a security configuration.

The Developer Security Guidelines will provide this awareness to the developer of GCCS software. The guidelines contained herein will be used by each software developer in the development of new software and in the enhancement and maintenance of legacy software. Software that is submitted through the GCCS configuration management (CM) processes will be required to adhere to these procedures and guidelines. The vehicle that will be used for testing compliance to these guidelines will be the Security Compliance (SeComp) tool. Software that fails to meet the criteria contained in the Developer Security Guidelines will be identified during the SeComp testing procedure and will be subject to rejection on that basis. Software that is known to not be in compliance with the Developer Security Guidelines must be justified in writing, stating the reasons for non-compliance and providing a schedule for achieving compliance.

# 2.0 SECURITY GUIDELINES

## 2.1 General Software Development Practices

This guideline is not an attempt to instruct the developer on the maintenance of a development cycle. The guidelines are provided to inform the developer of the required state of security developed into and behavior of software applications from an information security engineering perspective.

### 2.1.1 Getting Beyond Convenience Programming

Development and operations on a standalone, non-networked system, or workstation represent the most risk-free environment which might be encountered. A closed environment allows freedom in style and privilege, and often programming is strictly an exercise to accomplish a task or create a function which only the developer will use or understand. Other than concerns about damaging the personal work areas, there is little concern given to how a program might be used in the hands of someone else or in another environment. Casually developed programs can easily become dangerous to more structured and shared environments, either through unintentional, built-in threats, or the system configurations required by such a program.

Once a system or workstation becomes a shared entity, either through allowing shared physical access or allowing connections across a network, the risk environment changes drastically. Unintentional but damaging lines of code become destructive and the loose configurations forced by program runtime requirements render the system vulnerable to intrusion. Developing for shared environments not only adds to the complexity of the functional programs, but also extends the complexity of the risk environment where the small details must be given more thoughtful attention.

The first guideline is to overcome the urge to convenience program. Convenience programming will be touched upon in the following paragraphs, but most of all, it means to develop programs that make no assumptions about their runtime environment, use only well-documented application program interfaces (API), never trust to the integrity of the people or software that will execute the program, and most importantly, do not take shortcuts.

### 2.1.2 Permissions to the Group ID Level

The GCCS supports user account and profiling management mechanisms that enable the security administrators to control access to the GCCS application and database processes and objects. For these user account and profiling management mechanisms to work properly, the processes and objects on the file system must be maintained with cooperative permission sets that support the GCCS access control policy. The user account and profiling management mechanisms are centered around the control of access to the group membership. Therefore, controlling any access to a GCCS system object should consider a specific group membership, or profile combination of groups.

Under no circumstances will application or database programs be developed to take advantage of wholesale world permissions. Programs that require world permissions for access will be required to provide justification for the extent of access.

### 2.1.3 Undermining/Abusing the System and Security Configuration

The GCCS is permitted to operate by virtue of a strict adherence to the certified system security configuration. The GCCS certified configuration is described and defined in the GCCS TFM.

Under no circumstances will application or database programs be developed that modify the security configuration of the GCCS.

### 2.1.4 Un-initialized Variables

One of the most used techniques by the experienced hacker is to take advantage of un-initialized variables known to be within an executable program. Initialize each and every variable in a program, whether it is a binary executable or shell executable.

#### 2.1.4.1 Program Internals

Few programmers have thought about comments and structure since they left college. It is time to think about them again. The old story is true: maintenance happens. Try to ensure that, with a reasonable amount of experience, another programmer can read and understand the code without undue difficulty. Use meaningful variable names, do not sacrifice readability for convenience, and provide well commented documentation for clarification. Complex regular expressions in sed, for example, should be avoided. Do not use c-shell to program your scripts, use the bourne, korn, or posix shells. Do not pull software from Internet sources that you cannot explain, validate, and thoroughly document. Delivered executable software scripts and segment environments should not permit access to the system or any application without the appropriate identification and authentication (I&A) (i.e., no backdoor entries).

#### 2.1.4.2 Shell Environment

The shell is a great place to program quickly and conveniently, but it can be harmful if some precautions are not taken. Shell executables are obviously more capable of being hacked than binary executables.

There are several common shell environment variables that have been used to hack an executable to gain privilege. Again, this will not happen if these variables are initialized prior to being used in the program.

Under no circumstances will application or database programs be developed that require a privileged shell executable to maintain write permissions to the group or to the world.

### 2.1.5 Cleaning Up

Anything a program creates that is not intended to become a permanent object on the file system must be cleaned up before the program exits. It is even more important to pay attention to mode changes that may occur during the execution of a program. The developer must ensure that any mode changes are normalized before the program exits.

### 2.1.6 Sourcing Executables

Another well known hacking technique is to take advantage of relative pathnames used when calling an executable. All program calls to external executables must be sourced by referencing the absolute pathname location of the executable.

### 2.1.7 Error Checking

Check for return values after every command or call no matter how trivial. This is part of the responsibility that the developer shares in making each program=s execution predictable and reliable. Identify a strategy for exception or error indications that are returned and follow through with that strategy throughout the program. Error and exception messages should always be used to identify the real problem and where the problem was identified within the program. The program name will be included in the message.

### 2.1.8 Auditing Security-Relevant Activities

Applications that expect to operate on an operating system (OS) platform that executes with C2 class audit capabilities should take advantage of these audit capabilities. The audit subsystem provides API=s that can be used to record events in the system=s audit trail. It is recommended that all security-relevant events be recorded at a minimum.

## 2.2 Legacy Software Maintenance and Enhancements

The distinction between legacy and new software exists because of the legacy nature of the GCCS Common Operating Environment (COE) mission applications and database software. The expectation regarding legacy software is that it should be on a migration path toward Level 8 COE compliance, to include compliance with the security guidelines.

There are two approaches to achieving Level 8 COE, and therefore security, compliance. One way is to re-develop the application or database software programs to adhere to the compliance requirements. The second way is to re-engineer/modify the application or database software programs to adhere to the compliance requirements. It is anticipated that reaching full COE compliance in any regard will not happen quickly, therefore, in the interim it is expected that, from the security perspective, each application and database program developer will continually Atweak@ the product to conform to the security guidelines set forth in this document. Most importantly, it is required that the developers do not cause more violations once in possession of this guideline when providing enhanced or patched software programs.

## 2.3 New Software

New software developed for the GCCS will be required to meet the guidelines stated in this document. The fact that a software component is new development implies complete control over the software development process and therefore the enforcement of these security guidelines within a new software development is required and expected.

### 2.4 Design and Development Guidelines

The following sections provide guidance for the development process to ensure that software integrated into the GCCS does not violate or cause violation of the GCCS security policy.

### 2.4.1 Discretionary Access Control (DAC)

DAC permission sets should be controlled to the level of the group identification associated with the profile required to gain access to an application or database program. This means that world read, write, or execute permissions are not acceptable for the convenience of the program or programmer. For example, an application program that is accessed by users with the GCCS user profile should expect that GCCS group permission sets will suffice for any application operations. It is not acceptable to establish world permissions just in case they are required for application or user interactions. Additionally, all files must be owned by a valid user ID (UID).

#### 2.4.1.1 Umask Assignments

The process or user umask value causes a default permission set to be assigned to all files created by a process or user. Most Unix systems default to a umask setting that allows read and write (and execute for executable programs) permissions for the owner, group, and world. This default is not acceptable for a system that is attempting to control access to the file system objects.

Programs should <u>not</u> be written to assume a umask setting of any value. A program should be written such that it will expressly establish the permissions of a file created by the program as required by the program during normal usage. This may be done by creating the file and issuing the change mode command with the appropriate permission set.

There are numerous ways to establish the umask value for a process. One method is to insert the umask command and the umask value in the user home directory shell startup file. The *root* user is no different in this respect and the *root* user=s umask value can be assigned in the root file system level (A/@) shell startup file for the *root* user. It is not acceptable to establish or require a umask value for the *root* user that is anything but 077. A umask value of 077 will cause files to be created with a permission set of 600 (rw- --- ---) and directory permission of 700 (rwx --- ---). Only those files requiring execute permission should have the execute bit explicitly set. If permissions are to be granted to the group or world level, the program must explicitly establish these permissions after the file is created. This also infers that configuration files must not be required to assign a umask value that violates the umask requirement.

The current user umask setting that must be adhered to (as a minimum) in the GCCS is 002. This removes the world write access from the created file. The future direction for the umask is an assignment of 027, removing group write and world read, write, and execute permissions from a created file. Developers should strive to use a umask of 027 wherever possible.

#### 2.4.1.2 Data and Text File Permissions

Data and text files created by an application or database program should set the file permissions for what is required by its usage in the context of the program. For example, if a file is to be accessed by the GCCS group community, permissions should be established to the level of the group. If the file is only required to be read and never written or executed, the file should be established with read permission for the group only. If the file must be written to by the group members, the file should be established with write permissions for the group.

World permissions to GCCS application and database program data and text files are prohibited to the extent possible. If there are no known requirements for world permissions, no permission should be given. World write and execute access to a GCCS application or database program is not acceptable without clear justification. The normal acceptable permission set for data and text files is 440 (r-- r-- ---).

### 2.4.1.3 Shell Executable File Permissions

Shell executable file permissions should be set for what is required by its usage in the context of the program. For example, if a file is to be normally executed by the GCCS group community, read and execute permissions should be established to the level of the GCCS group.

World read and execute permissions to GCCS shell executable files are prohibited to the extent possible. If there are no known requirements for world permissions, no world permission should be given. Owner, group, or world write access to a GCCS shell executable file is not acceptable without clear justification. The normal acceptable permission set for a shell executable program is 550 (r-x r-x ---).

### 2.4.1.4 Binary Executable File Permissions

Binary executable file permissions should be set for what is required by its usage in the context of the program. For example, if a file is to be normally executed by an administrative group community, execute permissions should be established to the level of the administrative group.

World execute permissions to binary executable files are prohibited to the extent possible. If there are no known requirements for world execute permissions, no world permission should be given. World execute access to a GCCS binary executable file[1] is not acceptable without clear justification. The normal acceptable permission set for a GCCS binary executable program is 110 (--x --x ---).

### 2.4.1.5 Directory Permissions

Directory permissions should be set for what is required by its usage in the context of the programs that must access it. For example, if a directory is to be normally accessed by the GCCS group community, then read and execute permissions should be established to the level of the GCCS group. Write access may be provided if there is a need to add, rename, or remove entries from the directory.

World read, write, and execute permissions to GCCS directories are prohibited to the extent possible. If there are no known requirements for world permissions, no world permission should be given. World write access or especially world write and execute permissions to a GCCS directory is not acceptable without clear justification. The normal acceptable permission set for a GCCS directory is 770 (rwx rwx---), with a preferred permissions set of 750 (rwx r-x ---). Developers should strive to use the preferred permission set of 750 wherever possible.

[1] This does not include operating system binaries except where appropriate to control access to certain administrative executables.

#### 2.4.1.6 User Home Directory Permissions

User home directories are just that, user owned and maintained home directories. These directories should be allowed to be protected to the level of the user and not be restricted from this goal by an application or database program imposed requirement. There are better ways to implement the exception when information must be shared between users or between users and an application program.

Requiring world permissions of any kind to a user's home directory is not acceptable without clear justification. If at all possible, group permissions should be restricted to read and execute. The normal acceptable permission set for a user's home directory is 750 (rwx r-x ---), with a preferred permissions set of 700 (rwx --- ---). Developers should strive to adapt to use the preferred permission set of 700 wherever possible.

#### 2.4.1.7 System and Security File Permission Modifications

Certain directories in the Unix file system should be considered untouchable by application or database programs. These directories and their contents are not used by the general user community and are reserved to isolate administrative and security-related files and programs. A short list of directories and files in the category are:

*/etc*
*/etc/shadow*
*/etc/rc*
*/etc/mnttab*
*/var*
*/usr*

At most, these files may be required to be read by an application or database program. Under no circumstances will the permissions on system or security-related configuration files be modified by an application or database program. The acceptable permission set for files and programs in this category varies somewhat, however, the least restrictive set of permissions should be 444 (r-- r-- r--) with 440 (r-- r-- ---) preferred for non-executable files. Executable files in the */etc* directory will never allow world access. If this is required of an application file it should not be located in the */etc* directory.

### 2.4.2 Identification and Authentication

Without a doubt, the Identification and Authentication (I&A) file system components are the most important in the maintenance of a security configuration. The DAC permissions discussed in the previous section, are one aspect in the protection of the I&A information. Another and more directly felt aspect by the user community is the selection and maintenance of the individual password. The password provides the first line of defense against most[2] of the potential threats to a system.

#### 2.4.2.1 Password Database Entries

The password databases may vary in a Unix implementation, but their maintenance and protection do not. It

[2] Assuming that power or network components are susceptible to attack.

is often the case that an application or database program may add an application-oriented user account entry in the password file(s) on the system. This is not a problem provided that adequate notice is provided that the account password must be changed to reflect the construction required by the GCCS policy as stated in the GCCS TFM.

Under no circumstances will an application or database program require Agimme@ or easily guessed password, a blank or null password, or A+@ entry to be maintained within a password database or table. Additionally, when entries are added to the password file by an application or database program, they must provide valid information in all required password entry fields. In the case of the NIS+ password tables, the minimum of the first seven fields of the password account entry must be provided.

### 2.4.2.2 Transmission of Passwords

Application or database programs will not require that un-encrypted passwords be transmitted across local or wide area networks (LAN/WAN).

### 2.4.2.3 Sharing Password Entries

Application or database programs will not require that passwords be shared among user accounts.

### 2.4.2.4 Set User and Group ID Programs

Application or database programs will not require that set user ID (SUID) or set group ID (SGID) programs be given write access to the group or world user community. The AC@ shell will not be used to construct SUID or SGID programs. SUID and SGID programs will only be accessible to the least number of users required.

### 2.4.2.5 Usage of Root Privilege

The granting of the *root* user privilege will be restricted to administrative or security level tasks that can only be accomplished by the granting of that privilege. Under no circumstances will root privilege be granted for program/programmer convenience.

Application or database programs will not require that the *root* user ID be required for login across a LAN or WAN. This also infers that configuration files must not be required to set up this capability.

### 2.4.2.6 Use of Anonymous Accounts

Anonymous accounts that are required on the system must be capable of being configured in the following manner, and using the Joint Deployable Intelligence Support System (JDISS) application anonymous account as an example:

C Ensure that the anonymous user home directory is not owned by the anonymous user.

C Sanitize the */h/JDISS/data/share/etc/passwd* file to reflect only the ftp, anonymous, and share users (passwd).

C Add the unique group entry `ftpshare::150:` to the */h/JDISS/data/share/etc/group* file.

C Add the unique group entry `ftpshare::150:` to the */etc/group* file.

C Modify the */etc/passwd* entries for the ftp, anonymous, and share users to reflect the Aftpshare@ group.

C Execute `chgrp ftpshare` for the */h/JDISS/data/share/Audio*, *Documents*, *Images*, *Other*, *Text,* and *Video* directories.

C Execute `chmod 755 /h/JDISS/data/share.`

These actions effectively isolate an anonymous user to the anonymous hierarchy with no way to escape that environment.

#### 2.4.2.7 User IDs

Application or database programs will not require that UIDs be shared among user accounts nor will they require that a specific UID be associated with a user account.

#### 2.4.2.8 Group IDs

Application or database programs will not require that a specific group IDs (GIDs) be associated with a user or group account.

#### 2.4.2.9 Equivalency File (*.rhosts* and *equiv.hosts*) Usage

Application or database programs will <u>not</u> require:

C A specific hostname or user name be associated with an equivalency file entry. The spirit of this guideline is to prevent conflicts when the secure features of an information service (i.e., NIS+ netgroups) are implemented.

C A *.rhosts* file be required for the *root* user at the root file system (A/@) level.

C A *.rhosts* file must be given write access to the group or world communities.

C An equivalency file must contain a A++@ entry.

### 2.4.3 Audit

Under no circumstances will an application or database program require the use of or undermine the purpose of the audit files or capabilities of the system. The only allowable exception to this will be the capability of an application or database program to be able to write an audit record into the OS audit process to be entered into the audit trail.
Applications and database programs should consider the creation and use of application audit events for security-relevant activities that occur within the program. The creation and use of the application audit

events, and how to link these events to the audit daemon is a platform-specific process and must be engineered considering the platform's audit subsystem.

### 2.4.4 System Configuration

Under no circumstances will an application or database program undermine or substitute the purpose of or use of the system and security-related files and programs.

#### 2.4.4.1 Network Services

Under no circumstances will an application or database program require the use of a network service that subverts or undermines the security configuration of the system. Services that have been identified in this category are:

- finger,
- tftp,
- netstat,
- Unqualified cgi-bin programs,
- JAVA or JAVA-like programs that allow uncontrolled installation and operation of un-segmented and unqualified software programs, and
- rexec.

Under no circumstances will an application or database program require the use of a *.netrc* file. Additionally, no application or database program will require that a workstation be enabled in such a way to perform Internet Protocol (IP) packet forwarding functions.

The developers should avoid using the `rcp`, `rsh`, and `rlogin` commands wherever possible. In most cases, these commands can be replaced with more secure methods that accomplish the functions provided by these commands. Developers should clearly document the justification for the use of these commands.

#### 2.4.4.2 UUCP Usage

Under no circumstances will an application or database program require that the UUCP configuration be exported by an NFS server. Like any other login account, this account will be assigned a password constructed using the guidelines for a password selection and should never be required to be assigned a gimme, null, or shared password. The */usr/lib/uucp/Systems* file must be read/write protected to the owner (usually UUCP) only as this file contains the login information (including the un-encrypted passwords) required to log on to the remote systems. Therefore, an application or database program must not require less stringent permissions on this file.

Many versions of UUCP have been created over the years, and it is imperative that the developer does not implement an early version of the UUCP that have been shown to have gaping security holes. These security vulnerabilities range from subversion of the UUCP subsystem to the hijacking of root privilege. It is recommended that the developer follow the guidelines stated in the *Practical UNIX & Internet Security*, Second Edition, April 1996, by O'Reilly & Associates for the selection, configuration, and usage of the UUCP subsystem.

Finally, if UUCP is not going to be used, then it must be disabled by deleting the UUCP executables or modifying permissions to 0400 on all UUCP associated files and executables.

**2.4.4.3 NIS, NIS+ Usage**

The GCCS TFM Appendix B discusses the correct configuration and usage of the NIS and NIS+ services in the GCCS environment. Applications and database programs should be developed to take advantage of the NIS+ security features enabled in the GCCS environment. Under no circumstances will an application or database program require that these features be disabled or usurped. The following guidelines apply and must be considered:

- NIS+ high security modeusing the DESauth credentials is employed in the GCCS Solaris environment.

- NIS+ netgroups are used to control host access using the *rhosts, .hosts.equiv,* and *dfs* file system sharing.

- Master slave servers should not use NIS for password information.

- The */var/nis* directory and its contents will not be required to maintain write permissions to the group or world membership, and the contents should not be executable.

The GCCS TFM Appendix B also discusses the correct configuration and usage of the NIS services in the HP-UX combined with NIS+ services in the Solaris platforms in the GCCS environment. The developer must be familiar with the interactions between the two operating systems where NIS and NIS+ are concerned. The TFM Appendix B Section B2 covers the correct configuration of the NIS and NIS+ and the developer is directed to the GCCS TFM Appendix B for this information. Incorrect configuration of the HP-UX NIS and Solaris NIS+ can void the protections thought to be gained from their use.

**2.4.4.4 NFS and RPC Usage**

The NFS features of the GCCS are configured to take advantage of the security features offered by the NIS+ DESauth credentials. Under no circumstances will an application or database program require that settings that usurp this security (i.e., unqualified use of anon=0) be activated. The NIS, NIS+, NFS, and RPC commands all interact and involve dependencies on each other to affect the overall security of the network connectivity. These features must not be configured in such a way to usurp the security measures taken to control the network connectivity to the GCCS system. The developer is directed to the guidelines in the TFM Appendix B regarding the use of these features for the GCCS environments that utilize both the HP-UX and Solaris platforms.

## 3.0 SECOMP QUALITY ASSURANCE PROCEDURE

### 3.1 SeComp Testing Procedure

The SeComp testing software and procedure are described completely in the SeComp Administrators Manual. Basically, the SeComp software is executed three times:

C Immediately before the application segment is loaded.
C Immediately after the application segment is loaded.
C Immediately after the application has been operationally tested.

The first execution of SeComp sets up the environment where the application segment will be loaded and records the system's security state information. The second execution of the SeComp will detect if the system security configuration has been altered during the installation of the application segment. The third execution of the SeComp will detect if the system security configuration has been modified during the operation of the application segment.

### 3.2 Reporting Non-Compliant Software

As described in the SeComp documentation, an application segment that violates the documented security configuration for GCCS will be reviewed. The review will consider the risk factors to the system and could result in the application segment being returned to the developers to bring it into compliance.

## 4.0 KNOWN SOFTWARE DEFICIENCIES

### 4.1 Identifying and Justifying Security Non-Compliance

The risk associated with some non-compliant features of an application may be mitigated by simply being aware of the problem area. In these cases, it is possible that by performing some appropriate and controlled configuration modifications, the risk associated with the use of the feature may be sufficiently controlled. The key is having knowledge of the non-compliant feature or software.

It is imperative that any and all deviations from these guidelines be documented. A list of the deviations must accompany the software when it is delivered. The documentation must describe the deviation in detail, the justification for the deviation, and a schedule for bringing the application into compliance.

The Defense Information Systems Agency (DISA), along with the Joint Staff, will analyze any deviations from this guideline to determine the acceptability of risk. If the risk is deemed unacceptable, the software segment will be rejected and returned to the developer for correction.

## 5.0 CONCLUSIONS

Maintaining the GCCS security configuration is supported and dependent on the foundation of application and database software that comprises the GCCS. The security configuration will only be as effective as the operational software allows it to be.

It is the responsibility of each software development facility to become familiar with this document and the GCCS TFM, Security Policy, and Security Requirements for Automated Information Systems (AISs) documentation. In addition to the COE functional compliance, all software developed for the GCCS community must be compliant with the security guidelines and the spirit of the GCCS security policy. The security compliance of each application and database segment submitted is a testable commodity and will be tested in accordance with the guidance in this document as part of the segment verification and acceptance process.

## APPENDIX A

## REFERENCES

# APPENDIX A REFERENCES

Global Command and Control System Security Policy, Version 2.1, Draft, CJCSI 6731.01, April 30, 1996.

Department of Defense (DoD) Directive 5200.28, Security Requirements for Automated Information Systems, March 21, 1988.

Global Command and Control System Trusted Facility Manual, Version 2.1, September 30, 1996.

Practical UNIX & Internet Security, Second Edition, April 1996, by O'Reilly & Associates.